1. HBase简单读写流程?

读:

找到要读数据的region所在的RegionServer，然后按照以下顺序进行读取：先去BlockCache读取，若BlockCache没有，则到Memstore读取，若Memstore中没有，则到HFile中去读。

写:

找到要写数据的region所在的RegionServer，然后先将数据写到WAL(Write-Ahead Logging，预写日志系统)中，然后再将数据写到Memstore等待刷新，回复客户端写入完成。

2. 简述 HBase 的瓶颈

HBase 的瓶颈就是硬盘传输速度。HBase 的操作，它可以往数据里面 insert,也可以 update 一些数据，但 update 的实际上也是 insert，只是插入一个新的时间戳的一行。Delete 数据，也是 insert，只是 insert 一行带有 delete 标记的一行。
Hbase 的所有操作都是追加插入操作。Hbase 是一种日志集数据库。它的存储方式，像是日志文件一样。它是批量大量的往硬盘中写，通常都是以文件形式的读写。这个读写速度，就取决于硬盘与机器之间的传输有多快。而 Oracle 的瓶颈是硬盘寻道时间。它经常的操作时随机读写。要 update 一个数据，先要在硬盘中找到这个 block，然后把它读入内存，在内存中的缓存中修改，过段时间再回写回去。由于你寻找的 block 不同，这就存在一个随机的读。硬盘的寻道时间主要由转速来决定的。而寻道时间，技术基本没有改变，这就形成了寻道时间瓶颈。

3. 谈谈 HBase 集群安装注意事项?

需要注意的地方是 ZooKeeper 的配置。这与 hbase-env.sh 文件相关，文件中HBASE_MANAGES_ZK 环境变量用来设置是使用 hbase 默认自带的 Zookeeper 还是使用独立的 ZooKeeper。
HBASE_MANAGES_ZK=false 时使用独立的，为 true 时使用默认自带的。
某个节点的 HRegionServer 启动失败，这是由于这 3 个节点的系统时间不一致相差超过集群的检查时间 30s。

4. 怎样将 mysql 的数据导入到 hbase 中? 不能使用 sqoop，速度太慢了

A、一种可以加快批量写入速度的方法是通过预先创建一些空的 regions，这样当数据写入 HBase 时，会按照 region 分区情况，在集群内做数据的负载均衡。
B、hbase 里面有这样一个hfileoutputformat 类，他的实现可以将数据转换成hfile格式，通过 new 一个这个类，进行相关配置,这样会在 hdfs 下面产生一个文件，这个时候利用 hbase 提供的 jruby 的 loadtable.rb 脚本就可以进行批量导入。

5. HBase 接收数据，如果短时间导入数量过多的话就会被锁，该怎么办？ 集群数 16台，高可用性的环境。

参考：
通过调用 HTable.setAutoFlush(false)方法可以将 HTable 写客户端的自动 flush关闭，这样可以批量写入数据到 HBase，而不是有一条 put 就执行一次更新，只有当put 填满客户端写缓存时，才实际向 HBase 服务端发起写请求。默认情况下 auto flush是开启的。

6. 现在我们要对 Oracle 和 HBase 中的某些表进行更新，你是怎么操作?

disable '表名'
alter '表名', NAME => '列名', VERSIONS =>3
enable '表名'

7. hbase 的 rowkey 怎么创建好？列族怎么创建比较好?

hbase 存储时，数据按照 Row key 的字典序(byte order)排序存储。设计 key 时，要充分排序存储这个特性，将经常一起读取的行存储放到一起。(位置相关性) 一个列族在数据底层是一个文件，所以将经常一起查询的列放到一个列族中，列族尽量少，减少文件的寻址时间。

8. HBase 的 Insert 与 Update 的区别?

这个题目是就着最近的一次项目问的，当时实现的与 hbase 交互的三个方法分别为 insert、 delete、update。由于那个项目是对接的一个项目，对接的小伙伴和我协商了下，不将 update 合并为 insert，如果合并的话，按那个项目本身，其实通过 insert 执行 overwrite 相当于间接 地 Update，本质上，或者说在展现上是没什么区别的包括所调用的 put。但那仅仅是就着那 个项目的程序而言，如果基于 HBaseshell 层面。将同一 rowkey 的数据插入 HBase，其实虽 然展现一条，但是相应的 timestamp 是不一样的，而且最大的版本数可以通过配置文件进行 相应地设置。

9. Hbase 的实现原理

Hbase 的实现原理是 rpcProtocol。

10. hbase 过滤器实现原则

参考博文：https://zy19982004.iteye.com/blog/2088173

11. hbase 写数据的原理

1. 首先，Client 通过访问 ZK 来请求目标数据的地址。
2. ZK 中保存了-ROOT-表的地址，所以 ZK 通过访问-ROOT-表来请求数据地址。
3. 同样，-ROOT-表中保存的是.META.的信息，通过访问.META.表来获取具体的 RS。
4. .META.表查询到具体 RS 信息后返回具体 RS 地址给 Client。

Client 端获取到目标地址后，然后直接向该地址发送数据请求。

12. hbase 宕机了如何处理?

HBase 的 RegionServer 宕机超过一定时间后，HMaster 会将其所管理的 region 重新分布 到其他活动的 RegionServer 上，由于数据和日志都持久在 HDFS 中， 该操作不会导致数据丢失。所以数据的一致性和安全性是有保障的。 但是重新分配的 region 需要根据日志恢复原 RegionServer 中的内存 MemoryStore 表，这 会导致宕机的 region 在这段时间内无法对外提供服务。 而一旦重分布，宕机的节点重新启动后就相当于一个新的 RegionServer 加入集群，为了平 衡，需要再次将某些 region 分布到该 server。 因此，Region Server 的内存表 memstore 如何在节点间做到更高的可用，是 HBase 的一 个较大的挑战。

13. Hbase 中的 metastore 用来做什么的?

Hbase 的 metastore 是用来保存数据的，其中保存数据的方式有有三种第一种于第二种是本 地储存，第二种是远程储存这一种企业用的比较多。

14. hbase 客户端在客户端怎样优化?

Hbase 使用 JAVA 来运算的，索引 Java 的优化也适用于 hbase，在使用过滤器事记得开启 bloomfilter 可以是性能提高 3-4 倍，设置 HBASE_HEAPSIZE 设置大一些。

15. hbase 是怎样预分区的?

如何去进行预分区，可以采用下面三步： 1.取样，先随机生成一定数量的 rowkey,将取样数据按升序排序放到一个集合里 2.根据预分区的 region 个数，对整个集合平均分割，即是相关的 splitKeys. 3.HBaseAdmin.createTable(HTableDescriptor tableDescriptor,byte[][] splitkeys) 可以指定预分区的 splitKey，即是指定 region 间的 rowkey 临界值。

16. 怎样将 mysql 的数据导入到 hbase 中?

不能使用 sqoop，速度太慢了，提示如下： A、一种可以加快批量写入速度的方法是通过预先创建一些空的 regions，这样当 数据写入 HBase 时，会按照 region 分区情况，在集群内做数据的负载均衡。 B、hbase 里面有这样一个 hfileoutputformat 类，他的实现可以将数据转换成 hfile 格式，通过 new 一个这个类，进行相关配置,这样会在 hdfs 下面产生一个文件，这个 时候利用 hbase 提供的 jruby 的 loadtable.rb 脚本就可以进行批量导入。

17. 谈谈 HBase 集群安装注意事项?

需要注意的地方是 ZooKeeper 的配置。这与 hbase-env.sh 文件相关，文件中 HBASE_MANAGES_ZK 环境变量用来设置是使用 hbase 默认自带的 Zookeeper 还 是使用独立的 ZooKeeper。 HBASE_MANAGES_ZK=false 时使用独立的，为 true 时 使用默认自带的。 某个节点的 HRegionServer 启动失败，这是由于这 3 个节点的系统时间不一致相 差超过集群的检查时间 30s。

18. hive跟hbase的区别是?

1. 两者分别是什么?
   Apache Hive是一个构建在Hadoop基础设施之上的数据仓库。通过Hive可以使用HQL语言查询存放在HDFS上的数据。HQL是一种类SQL语言，这种语言最终被转化为Map/Reduce. 虽然Hive提供了SQL查询功能，但是Hive不能够进行交互查询--因为它只能够在Haoop上批量的执行Hadoop。
   Apache HBase是一种Key/Value系统，它运行在HDFS之上。和Hive不一样，Hbase的能够在它的数据库上实时运行，而不是运行MapReduce任务。Hive被分区为表格，表格又被进一步分割为列簇。列簇必须使用schema定义，列簇将某一类型列集合起来（列不要求schema定义）。例如，

“message”列簇可能包含：“to”, ”from” “date”, “subject”, 和”body”. 每一个 key/value对在Hbase中被定义为一个cell，每一个key由row-key，列簇、列和时间戳。在Hbase中，行是key/value映射的集合，这个映射通过row-key来唯一标识。Hbase利用Hadoop的基础设施，可以利用通用的设备进行水平的扩展。

2. 两者的特点
   Hive帮助熟悉SQL的人运行MapReduce任务。因为它是JDBC兼容的，同时，它也能够和现存的SQL工具整合在一起。运行Hive查询会花费很长时间，因为它会默认遍历表中所有的数据。虽然有这样的缺点，一次遍历的数据量可以通过Hive的分区机制来控制。分区允许在数据集上运行过滤查询，这些数据集存储在不同的文件夹内，查询的时候只遍历指定文件夹（分区）中的数据。这种机制可以用来，例如，只处理在某一个时间范围内的文件，只要这些文件名中包括了时间格式。
   HBase通过存储key/value来工作。它支持四种主要的操作：增加或者更新行，查看一个范围内的cell，获取指定的行，删除指定的行、列或者是列的版本。版本信息用来获取历史数据（每一行的历史数据可以被删除，然后通过Hbase compactions就可以释放出空间）。虽然HBase包括表格，但是schema仅仅被表格和列簇所要求，列不需要schema。Hbase的表格包括增加/计数功能。
3. 限制
   Hive目前不支持更新操作。另外，由于hive在hadoop上运行批量操作，它需要花费很长的时间，通常是几分钟到几个小时才可以获取到查询的结果。Hive必须提供预先定义好的schema将文件和目录映射到列，并且Hive与ACID不兼容。
   HBase查询是通过特定的语言来编写的，这种语言需要重新学习。类SQL的功能可以通过Apache Phonenix实现，但这是以必须提供schema为代价的。另外，Hbase也并不是兼容所有的ACID特性，虽然它支持某些特性。最后但不是最重要的--为了运行Hbase，Zookeeper是必须的，zookeeper是一个用来进行分布式协调的服务，这些服务包括配置服务，维护元信息和命名空间服务。
4. 应用场景
   Hive适合用来对一段时间内的数据进行分析查询，例如，用来计算趋势或者网站的日志。Hive不应该用来进行实时的查询。因为它需要很长时间才可以返回结果。
   Hbase非常适合用来进行大数据的实时查询。Facebook用Hbase进行消息和实时的分析。它也可以用来统计Facebook的连接数。

总结
Hive和Hbase是两种基于Hadoop的不同技术--Hive是一种类SQL的引擎，并且运行MapReduce任务，Hbase是一种在Hadoop之上的NoSQL 的Key/value数据库。当然，这两种工具是可以同时使用的。就像用Google来搜索，用FaceBook进行社交一样，Hive可以用来进行统计查询，HBase可以用来进行实时查询，数据也可以从Hive写到Hbase，设置再从Hbase写回Hive。

19. Hbase 内部是什么机制?

参考：https://blog.csdn.net/zhanaolu4821/article/details/82019442

20. flush 的过程?

为了减少flush过程对读写的影响，HBase采用了类似于两阶段提交的方式，将整个flush过程分为三个阶段：
prepare阶段：遍历当前Region中的所有Memstore，将Memstore中当前数据集kvset做一个快照snapshot，然后再新建一个新的kvset。后期的所有写入操作都会写入新的kvset中，而整个flush阶段读操作会首先分别遍历kvset和snapshot，如果查找不到再会到HFile中查找。prepare阶段需要加一把updateLock对写请求阻塞，结束之后会释放该锁。因为此阶段没有任何费时操作，因此持锁时间很短。
flush阶段：遍历所有Memstore，将prepare阶段生成的snapshot持久化为临时文件，临时文件会统一放到目录.tmp下。这个过程因为涉及到磁盘IO操作，因此相对比较耗时。

commit阶段：遍历所有的Memstore，将flush阶段生成的临时文件移到指定的ColumnFamily目录下，针对HFile生成对应的storefile和Reader，把storefile添加到HStore的storefiles列表中，最后再清空prepare阶段生成的snapshot。
上述flush流程可以通过日志信息查看：
/*** prepare阶段 ****/
2016-02-04 03:32:41,516 INFO [MemStoreFlusher.1] regionserver.HRegion: Started memstore flush for sentry_sgroup1_data,{\xD4\x00\x00\x01|\x00\x00\x03\x82\x00\x00\x00?\x06\xDA`\x13\xCAE\xD3C\xA3:*1\xD6\x99:\x88\x7F\xAA*\xD6[L\xF0\x92\xA6\xFB^\xC7\xA4\xC7\xD7\x8Fv\xCAT\xD2\xAF,1452217805884.572ddf0e8cf0b11aee2273a95bd07879., current region memstore size 128.9 M

/*** flush阶段 ****/
2016-02-04 03:32:42,423 INFO [MemStoreFlusher.1] regionserver.DefaultStoreFlusher: Flushed, sequenceid=1726212642, memsize=128.9 M, hasBloomFilter=true, into tmp file hdfs://hbase1/hbase/data/default/sentry_sgroup1_data/572ddf0e8cf0b11aee2273a95bd07879/.tmp/021a430940244993a9450dccdfdcb91d

/*** commit阶段 ****/
2016-02-04 03:32:42,464 INFO [MemStoreFlusher.1] regionserver.HStore: Added hdfs://hbase1/hbase/data/default/sentry_sgroup1_data/572ddf0e8cf0b11aee2273a95bd07879/d/021a430940244993a9450dccdfdcb91d, entries=643656, sequenceid=1726212642, filesize=7.1 M

21. HBase 在进行模型设计时重点在什么地方？一张表中定义多少个 Column Family最合适？为什么？

具体看表的数据，一般来说划分标准是根据数据访问频度，如一张表里有些列访问相对频繁，而另一些列访问很少，这时可以把这张表划分成两个列族，分开存储，提高访问效率。

22. 请采取尽量多的方式将MySQL数据导入到HBase中（至少三种方式），并描述各种方式的优缺点。

1、Put API
Put API可能是将数据快速导入HBase表的最直接的方法。但是在导入【大量数据】时不建议使用！但是可以作为简单数据迁移的选择，直接写个代码批量处理，开发简单、方便、可控强。
2、MapReduce Job
推荐使用sqoop，它的底层实现是mapreduce，数据并行导入的，这样无须自己开发代码，过滤条件通过query参数可以实现。
Sqoop是一款开源的工具，主要用于在Hadoop(Hive)与传统的数据库(mysql、postgresql...)间进行数据的传递，可以将MySQL中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到Mysql中。
参考Index of /docs。
采用如下命令：sqoop import
--connect jdbc:mysql://localhost/db
--username root -P
--table mysql_order
--columns "id,name"
--hbase-table hbase_order
--column-family f

--hbase-row-key id
--query "select id,name from mysql_order where..."
-m 1
3、采用Bulk load装载数据
bulk-load的作用是用mapreduce的方式将hdfs上的文件装载到hbase中，对于海量数据装载入hbase非常有用。
需要将MySQL的表数据导出为TSV格式（因为后面使用Import TSV工具），还需要确保有一个字段可以表示HBase表行的row key。

23. 在每天增量数据较大（每天大约5T左右）时，在设计表和Hbase整体的参数配置方面有何建议?

Rowkey的设计，遵从几个原则，在长度方面在满足需求的情况下越短越好，因为数据在持久化文件Hfile中是按照keyvalue存储的，如果rowkey过长，数据量大的是后光rowkey就要占据很大空间，影响存储效率，第二个是满足散列原则，避免数据热点堆积现象的发生，还有必须保证rowkey的唯一性，并且覆盖尽可能多的业务场景
参数：hbase.regionserver.handler.count：rpc请求的线程数量，默认值是10，生产环境建议使用100，也不是越大越好，特别是当请求内容很大的时候，比如scan/put几M的数据，会占用过多的内存，有可能导致频繁的GC，甚至出现内存溢出。
hbase.hregion.max.filesize：默认是10G， 如果任何一个column familiy里的StoreFile超过这个值, 那么这个Region会一分为二，因为region分裂会有短暂的region下线时间(通常在5s以内)，为减少对业务端的影响，建议手动定时分裂，可以设置为60G。
hbase.hstore.compaction.max：默认值为10,一次最多合并多少个storefile，避免OOM。

24. hbase region 多大会分区，Spark 读取 hbase 数据是如何划分 partition 的?

答：region 超过了 hbase.hregion.max.filesize 这个参数配置的大小就会自动裂分，默认 值是 1G。 默认情况下，hbase 有多少个 region，Spark 读取时就会有多少个 partition。

25. Hbase行健列族的概念，物理模型，表的设计原则?

行健：是hbase表自带的，每个行健对应一条数据。
列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节 数组，其中的数据可以有很多，通过时间戳来区分。
物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节 点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始 点，-ROOT又可以存储.META的起始点。
rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入。
regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16 字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。
列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经 常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

26. HBase的Insert与Update的区别?

这个题目是就着最近的一次项目问的，当时实现的与hbase交互的三个方法分别为insert、 delete、update。由于那个项目是对接的一个项目，对接的小伙伴和我协商了下，不将update合 并为insert，如果合并的话，按那个项目本身，其实通过insert执行overwrite相当于间接地 Update，本质上，或者说在展现上是没什么区别的包括所调用的put。但那仅仅是就着那个项目 的程序而言，如果基于HBase shell层面。将同一rowkey的数据插入HBase，其实虽然展现一条， 但是相应的timestamp是不一样的，而且最大的版本数可以通过配置文件进行相应地设置。

27. HBase 的特点是什么

(1) Hbase一个分布式的基于列式存储的数据库,基于Hadoop的hdfs存储，zookeeper进行管理。

(2) Hbase适合存储半结构化或非结构化数据，对于数据结构字段不够确定或者杂乱无章很难按一个概念去抽取的数据。

(3) Hbase为null的记录不会被存储.

(4)基于的表包含rowkey，时间戳，和列族。新写入数据时，时间戳更新，同时可以查询到以前的版本.

(5) hbase是主从架构。hmaster作为主节点，hregionserver作为从节点。

28. hbase如何导入数据?

1. 通过HBase API进行批量写入数据；
2. 使用Sqoop工具批量导数到HBase集群；
3. 使用MapReduce批量导入；
4. HBase BulkLoad的方式。

29. Hbase和hive 有什么区别

Hive和Hbase是两种基于Hadoop的不同技术--Hive是一种类SQL 的引擎，并且运行MapReduce 任务，Hbase 是一种在Hadoop之上的NoSQL的Key/vale数据库。当然，这两种工具是可以同时使用的。就像用Google 来搜索，用FaceBook 进行社交一样，Hive 可以用来进行统计查询，HBase 可以用来进行实时查询，数据也可以从Hive 写到Hbase，设置再从Hbase 写回Hive。

Hive是一个构建在Hadoop 基础之上的数据仓库。通过Hive可以使用HQL语言查询存放在HDFS 上的数据。

HQL是一种类SQL语言，这种语言最终被转化为Map/Reduce. 虽然Hive提供了SQL查询功能，但是Hive不能够进行交互查询,因为它只能够在Haoop上批量的执行Hadoop。

Hive 被分区为表格，表格又被进一步分割为列簇。列簇必须使用schema 定义，列簇将某一类型列集合起来（列不要求schema定义）。

限制：

Hive 目前不支持更新操作。

另外，由于hive在hadoop上运行批量操作，它需要花费很长的时间，通常是几分钟到几个小时才可以获取到查询的结果。

Hive 适合用来对一段时间内的数据进行分析查询，例如，用来计算趋势或者网站的日志。

Hive 不应该用来进行实时的查询。因为它需要很长时间才可以返回结果。

HBase 查询是通过特定的语言来编写的，这种语言需要重新学习。类SQL 的功能可以通过Apache Phonenix 实现，但这是以必须提供schema 为代价的。另外，Hbase 也并不是兼容所有的ACID 特性，虽然它支持某些特性。最后但不是最重要的--为了运行Hbase，Zookeeper是必须的，zookeeper 是一个用来进行分布式协调的服务，这些服务包括配置服务，维护元信息和命名空间服务。

Hbase非常适合用来进行大数据的实时查询。Facebook用Hbase 进行消息和实时的分析。它也可以用来统计Facebook的连接数。

HBase 是一种Key/Value 系统，它运行在HDFS 之上。和Hive 不一样，Hbase 的能够在

它的数据库上实时运行，而不是运行MapReduce 任务。

30. 描述Hbase的rowKey的设计原则

- Rowkey长度原则

Rowkey 是一个二进制码流，Rowkey 的长度被很多开发者建议说设计在10~100 个字节，不过建议是越短越好，不要超过16 个字节。

原因如下：

（1）数据的持久化文件HFile 中是按照KeyValue 存储的，如果Rowkey 过长比如100 个字节，1000 万列数据光Rowkey 就要占用100*1000 万=10 亿个字节，将近1G 数据，这会极

大影响HFile 的存储效率；

（2）MemStore 将缓存部分数据到内存，如果Rowkey 字段过长内存的有效利用率会降低，系统将无法缓存更多的数据，这会降低检索效率。因此Rowkey 的字节长度越短越好。

（3）目前操作系统是都是64 位系统，内存8 字节对齐。控制在16 个字节，8 字节的整数倍利用操作系统的最佳特性。

- Rowkey散列原则

如果Rowkey 是按时间戳的方式递增，不要将时间放在二进制码的前面，建议将Rowkey的高位作为散列字段，由程序循环生成，低位放时间字段，这样将提高数据均衡分布在每个Regionserver 实现负载均衡的几率。如果没有散列字段，首字段直接是时间信息将产生所有新数据都在一个 RegionServer 上堆积的热点现象，这样在做数据检索的时候负载将会集中在个别RegionServer，降低查询效率。

- Rowkey唯一原则

必须在设计上保证其唯一性。

31. 描述 Hbase 的 rowKey 的设计原则

联系 region 和 rowkey 关系说明，设计可参考以下三个原则.

rowkey 长度原则
rowkey 是一个二进制码流，可以是任意字符串，最大长度 64kb，实际应用中一般为 10-100bytes，以 byte[] 形式保存，一般设计成定长。建议越短越好，不要超过 16 个字节， 原因如下：

数据的持久化文件 HFile 中是按照 KeyValue 存储的，如果 rowkey 过长会极大影响 HFile 的存储效率
MemStore 将缓存部分数据到内存，如果 rowkey 字段过长，内存的有效利用率就会降低，系统不能缓存更多的数据，这样会降低检索效率

rowkey 散列原则
如果 rowkey 按照时间戳的方式递增，不要将时间放在二进制码的前面，建议将 rowkey 的高位作为散列字段，由程序随机生成，低位放时间字段，这样将提高数据均衡分布在每个 RegionServer，以实现负载均衡的几率。如果没有散列字段，首字段直接是时间信息，所有的数据都会集中在一个 RegionServer 上，这样在数据检索的时候负载会集中在个别的 RegionServer 上，造成热点问题，会降低查询效率。

rowkey 唯一原则
必须在设计上保证其唯一性，rowkey 是按照字典顺序排序存储的，因此， 设计 rowkey 的时候，要充分利用这个排序的特点，将经常读取的数据存储到一块，将最近可能会被访问的数据放到一块。

32. 描述Hbase中scan和get的功能以及实现的异同.

HBase的查询实现只提供两种方式：

1、按指定RowKey 获取唯一一条记录，get方法（org.apache.hadoop.hbase.client.Get）

Get 的方法处理分两种 : 设置了ClosestRowBefore 和没有设置的rowlock .主要是用来保证行的事务性，即每个get 是以一个row 来标记的.一个row中可以有很多family 和column.

2、按指定的条件获取一批记录，scan方法(org.apache.Hadoop.hbase.client.Scan）实现条件查询功能使用的就是scan 方式.

1)scan 可以通过setCaching 与setBatch 方法提高速度(以空间换时间)；

2)scan 可以通过setStartRow 与setEndRow 来限定范围([start，end)start 是闭区间，

end 是开区间)。范围越小，性能越高。

3)、scan 可以通过setFilter 方法添加过滤器，这也是分页、多条件查询的基础。

33. 请描述Hbase中scan对象的setCache和setBatch 方法的使用.

为设置获取记录的列个数，默认无限制，也就是返回所有的列.每次从服务器端读取的行数，默认为配置文件中设置的值.

34. 请详细描述Hbase中一个Cell 的结构

HBase 中通过row 和columns 确定的为一个存贮单元称为cell。

Cell：由{row key, column(= +), version}唯一确定的单元。cell 中的数

据是没有类型的，全部是字节码形式存贮。

35. 请描述如何解决Hbase中region太小和region太大带来的冲突.

Region过大会发生多次compaction，将数据读一遍并重写一遍到hdfs 上，占用io，region过小会造成多次split，region 会下线，影响访问服务，调整hbase.hregion.max.filesize 为256m.

36. 以 start-hbase.sh 为起点，Hbase 启动的流程是什么?

start-hbase.sh 的流程如下：

1.运行 hbase-config.sh

hbase-config.sh的作用：

1>.装载相关配置，如HBASE_HOME目录，conf目录，regionserver机器列表，JAVA_HOME 目录等，它会调用$HBASE_HOME/conf/hbase-env.sh .

2>.解析参数（0.96 版本及以后才可以带唯一参数 autorestart，作用就是重启）

3>.调用 hbase-daemon.sh 来启动 master.

4>.调用 hbase-daemons.sh 来启动 regionserver zookeeper master-backup.

2.hbase-env.sh 的作用：

主要是配置 JVM 及其 GC 参数，还可以配置 log 目录及参数，配置是否需要 hbase 管

理 ZK，配置进程 id 目录等.

3.hbase-daemons.sh 的作用：根据需要启动的进程，

如 zookeeper,则调用 zookeepers.sh

如 regionserver，则调用 regionservers.sh

如 master-backup，则调用 master-backup.sh

4.zookeepers.sh 的作用：

如果 hbase-env.sh 中的 HBASE_MANAGES_ZK"="true"，那么通过ZKServerTool这个类解析xml配置文件，获取 ZK 节点列表，然后通过 SSH 向这些节点发送远程命令执行。

5.regionservers.sh 的作用：

与 zookeepers.sh 类似，通过配置文件，获取 regionserver 机器列表，然后 SSH 向这些机器发送远程命令：

6.master-backup.sh 的作用：

通过 backup-masters 这个配置文件，获取 backup-masters 机器列表,然后 SSH 向这些机器发送远程命令。

37. 简述 HBASE中compact用途是什么，什么时候触发，分为哪两种,有什么区别，有哪些相关配置参数?

在hbase中每当有memstore数据flush到磁盘之后，就形成一个storefile，当storeFile的数量达到一定程度后，就需要将 storefile 文件来进行 compaction 操作。

Compact 的作用：

1>.合并文件

2>.清除过期，多余版本的数据

3>.提高读写数据的效率

HBase 中实现了两种 compaction 的方式：minor and major. 这两种 compaction 方式的区别是：

1、Minor 操作只用来做部分文件的合并操作以及包括 minVersion=0 并且设置 ttl 的过

期版本清理，不做任何删除数据、多版本数据的清理工作。

2、Major 操作是对 Region 下的HStore下的所有StoreFile执行合并操作，最终的结果是整理合并出一个文件。

38. HBase 中实现了两种 compaction 的方式： minor and major这两种compaction方式的区别是：

Minor 操作：只合并小文件，对TTL过期数据设置过期清理，不会对文件内容进行清除操作
Major 操作：对 Region 下同一个 Column family 的 StoreFile 合并为一个大文件，并且清除删除、过期、多余版本的数据

39. 简述 Hbase filter 的实现原理是什么？结合实际项目经验，写出几个使用filter 的场景。

HBase 为筛选数据提供了一组过滤器，通过这个过滤器可以在 HBase 中的数据的多个维度（行，列，数据版本）上进行对数据的筛选操作，也就是说过滤器最终能够筛选的数据能够细化到具体的一个存储单元格上（由行键， 列名，时间戳定位）。

RowFilter、PrefixFilter。hbase 的 filter 是通过 scan 设置的，所以是基于 scan 的查询结果进行过滤.过滤器的类型很多，但是可以分为两大类——比较过滤器，专用过滤器。过滤器的作用是在服务端判断数据是否满足条件，然后只将满足条件的数据返回给客户端；如在进行订单开发的时候，我们使用 rowkeyfilter 过滤出某个用户的所有订单。

40. 简述HBase的瓶颈

提示：HBase的瓶颈就是硬传输速度，Hbase 的操作，它可以往数据里面 insert，也可以update一些数据，但update 的实际上也是insert，只是插入一个新的时间戳的一行，delete数据，也是insert，只是insert一行带有delete标记的一行。hbase的所有操作都是追加插入操作。hbase是一种日志集数据库。它的存储方式，像是日志文件一样。它是批量大量的往硬盘中写，通常都是以文件形式的读写。这个读写速度，就取决于硬盘与机器之间的传输有多快。而oracle的瓶颈是硬盘寻到时间。它经常的操作时随机读写。要update一个数据，先要在硬盘中找到这个block，然后把它读入内存，在内存中的缓存中修改，过段时间再回写回去。由于你寻找的block不通，这就存在一个随机的读。硬盘的寻道时间主要由转速来决定。而寻道时间，技术基本没有改变，这就形成了寻道时间瓶颈。

41. HBase来源于哪篇博文？ C

A The Google File System
B MapReduce
C BigTable
D Chubby

42. 下面对HBase的描述哪些是正确的？ B、C、D

A 不是开源的
B 是面向列的
C 是分布式的
D 是一种NoSQL数据库

43. HBase依靠（）存储底层数据 A

A HDFS
B Hadoop
C Memory
D MapReduce

44. HBase依赖（）提供消息通信机制 A

A Zookeeper
B Chubby
C RPC
D Socket

45. HBase依赖（）提供强大的计算能力 D

A Zookeeper
B Chubby
C RPC
D MapReduce

46. MapReduce与HBase的关系，哪些描述是正确的？ B、C

A 两者不可或缺，MapReduce是HBase可以正常运行的保证
B 两者不是强关联关系，没有MapReduce，HBase可以正常运行
C MapReduce可以直接访问HBase
D 它们之间没有任何关系

47. 下面哪些选项正确描述了HBase的特性？ A、B、C、D

A 高可靠性
B 高性能
C 面向列
D可伸缩

48. 下面与Zookeeper类似的框架是？ D

A Protobuf
B Java
C Kafka
（Kafka是一个高吞吐量分布式消息系统。linkedin开源的kafka。 Kafka就跟这个名字一样，设计非常独特。首先，kafka的开发者们认为不需要在内存里缓存什么数据，操作系统的文件缓存已经足够完善和强大，只要你不搞随机写，顺序读写的性能是非常高效的。kafka的数据只会顺序append，数据的删除策略是累积到一定程度或者超过一定时间再删除。Kafka另一个独特的地方是将消费者信息保存在客户端而不是MQ服务器，这样服务器就不用记录消息的投递过程，每个客户端都自己知道自己下一次应该从什么地方什么位置读取消息，消息的投递过程也是采用客户端主动pull的模型，这样大大减轻了服务器的负担。Kafka还强调减少数据的序列化和拷贝开销，它会将一些消息组织成Message Set做批量存储和发送，并且客户端在pull数据的时候，尽量以zero-copy的方式传输，利用sendfile（对应java里的 FileChannel.transferTo/transferFrom）这样的高级IO函数来减少拷贝开销。可见，kafka是一个精心设计，特定于某些应用的MQ系统，这种偏向特定领域的MQ系统我估计会越来越多，垂直化的产品策略值的考虑）
D Chubby
（MapReduce 很多人已经知道了，但关于Chubyy似乎熟悉它的就非常有限，这倒是不奇怪，因为MapReduce是一个针对开发人员的 ProgrammingModel，自然会有很多人去学习它，而Chubby更多的是一种为了实现MapReduce或者Bigtable而构建的内部的 工具，对于开发人员来说基本上是透明的。Chubby首先是一个分布式的文件系统。Chubby能够提供机制使得client可以在Chubby service上创建文件和执行一些文件的基本操作。说它是分布式的文件系统，是因为一个Chubby cell是一个分布式的系统，一般包含了5台机器，整个文件系统是部署在这5台机器上的。但是，从更高一点的语义层面上，Chubby是一个 lock service，一个针对松耦合的分布式系统的lock service。所谓lock service，就是这个service能够提供开发人员经常用的“锁”，“解锁”功能。通过Chubby，一个分布式系统中的上千个

client都能够 对于某项资源进行“加锁”，“解锁”。那么，Chubby是怎样实现这样的“锁”功能的？就是通过文件。Chubby中的“锁”就是文件，在上例 中，创建文件其实就是进行“加锁”操作，创建文件成功的那个server其实就是抢占到了“锁”。用户通过打开、关闭和读取文件，获取共享锁或者独占锁； 并且通过通信机制，向用户发送更新信息。

综上所述，Chubby是一个lock service，通过这个lock service可以解决分布式中的一致性问题，而这个lock service的实现是一个分布式的文件系统。）

49. 下面与HDFS类似的框架是？C

A NTFS
B FAT32
C GFS(也是分布式文件系统，谷歌自己的分布式文件系统)
D EXT3

50. 下面哪些概念是HBase框架中使用的？A、C

A HDFS
B GridFS
C Zookeeper
D EXT3

51. LSM含义是？A

A 日志结构合并树（Log-Structured Merge Tree）
B 二叉树
C 平衡二叉树
D 长平衡二叉树

52. 下面对LSM结构描述正确的是？ A、C

A 顺序存储
B 直接写硬盘
C 需要将数据Flush到磁盘
D 是一种搜索平衡树

53. LSM更能保证哪种操作的性能？B

A 读
B 写
C 随机读
D 合并

54. LSM的读操作和写操作是独立的？A

A 是。
B 否。
C LSM并不区分读和写
D LSM中读写是同一种操作

55. LSM结构的数据首先存储在（）。 B

A 硬盘上
B 内存中
C 磁盘阵列中
D 闪存中

56. HFile数据格式中的Data字段用于（）。A

A 存储实际的KeyValue数据
B 存储数据的起点
C 指定字段的长度
D 存储数据块的起点

57. HFile数据格式中的MetaIndex字段用于（）。D

A Meta块的长度
B Meta块的结束点
C Meta块数据内容
D Meta块的起始点

58. HFile数据格式中的Magic字段用于（）。A

A 存储随机数，防止数据损坏
B 存储数据的起点
C 存储数据块的起点
D 指定字段的长度

59. HFile数据格式中的KeyValue数据格式，下列选项描述正确的是（）。A、D

A 是byte[]数组
B 没有固定的结构
C 数据的大小是定长的
D 有固定的结构

60. HFile数据格式中的KeyValue数据格式中Value部分是（）。C

A 拥有复杂结构的字符串
B 字符串
C 二进制数据
D 压缩数据
第三部分：HBase高级应用介绍

61. HBase中的批量加载底层使用（）实现。A

A MapReduce
B Hive
C Coprocessor
D Bloom Filter

62. HBase性能优化包含下面的哪些选项？A、B、C、D

A 读优化
B 写优化
C 配置优化
D JVM优化

63. Rowkey设计的原则，下列哪些选项的描述是正确的？A、B、C

A 尽量保证越短越好
B 可以使用汉字
C 可以使用字符串
D 本身是无序的

64. HBase构建二级索引的实现方式有哪些？A、B

A MapReduce
B Coprocessor
(HBase在0.92之后引入了协处理器(coprocessors)，实现一些激动人心的新特性：能够轻易建立二次索引、复杂过滤器(谓词下推)以及访问控制等)
C Bloom Filter
D Filter

65. 关于HBase二级索引的描述，哪些是正确的？A、B

A 核心是倒排表
B 二级索引概念是对应Rowkey这个“一级”索引
C 二级索引使用平衡二叉树
D 二级索引使用LSM结构

66. 下列关于Bloom Filter的描述正确的是？A、C

A 是一个很长的二进制向量和一系列随机映射函数
B 没有误算率
C 有一定的误算率
D 可以在Bloom Filter中删除元素
第四部分：HBase安装、部署、启动

67. HBase官方版本可以安装在什么操作系统上？A、B、C

A CentOS
B Ubuntu
C RedHat
D Windows

68. HBase虚拟分布式模式需要（）个节点？A

A 1
B 2
C 3
D 最少3个

69. HBase分布式模式最好需要（）个节点？C

A 1
B 2
C 3
D 最少

70. 下列哪些选项是安装HBase前所必须安装的？A、B

A 操作系统
B JDK
C Shell Script
D Java Code

71. 解压.tar.gz结尾的HBase压缩包使用的Linux命令是？A

A tar -zxvf
B tar -zx
C tar -s
D tar -nf

72. HBase 的存储结构?

Hbase 中的每张表都通过行键 (rowkey) 按照一定的范围被分割成多个子表（HRegion），默认一个 HRegion 超过 256M 就要被分割成两个，由 HRegionServer 管理，管理哪些 HRegion 由 Hmaster 分配。 HRegion 存取一个子表时，会创建一个 HRegion 对象，然后对表的每个列族 （Column Family）创建一个 store 实例， 每个 store 都会有 0个或多个 StoreFile 与之对应，每个 StoreFile 都会对应一个 HFile， HFile 就是实际的存储文件，因此，一个 HRegion 还拥有一个 MemStore 实例。

73. Hbase 和 hive 有什么区别？hive 与 hbase 的底层存储是什么？hive 是产生的原因是什么？habase 是为了弥补 hadoop 的什么缺陷?

共同点：

hbase与hive都是架构在hadoop之上的。都是用hadoop作为底层存储。
区别：

Hive是建立在Hadoop之上为了减少MapReducejobs编写工作的批处理系统，HBase是为了支持弥补Hadoop对实时操作的缺陷的项目 。
想象你在操作RMDB数据库，如果是全表扫描，就用Hive+Hadoop，如果是索引访问，就用HBase+Hadoop；
Hive query就是MapReduce jobs可以从5分钟到数小时不止，HBase是非常高效的，肯定比Hive高效的多；
Hive本身不存储和计算数据，它完全依赖于 HDFS 和 MapReduce，Hive中的表纯逻辑；
hive借用hadoop的MapReduce来完成一些hive中的命令的执行；
hbase是物理表，不是逻辑表，提供一个超大的内存hash表，搜索引擎通过它来存储索引，方便查询操作；
hbase是列存储；
hdfs 作为底层存储，hdfs 是存放文件的系统，而 Hbase 负责组织文件；
hive 需要用到 hdfs 存储文件，需要用到 MapReduce 计算框架。

74. 说一说Hbase和Hive

这两个东西更是扯远了，但也可以聊一聊：

本质来说，Hive实质上是MapReduce作业的作业流，减少了MapReduce作业的开发成本，通过把数据映射成为hive中的表结构，可以轻松地通过类SQL语言，即HQL来完成查询作业，开发成本较低，深受大家欢迎

而Hbase本质上只是数据库而已，其执行过程，完全没有MapReduce作业，不然，hbase的速度得慢成什么样子：

就操作而言，Hive只能查询数据，而无法更新数据，这也是MapReduce作业的特点，只能对固有的数据进行分析；而Hbase则是可以对数据进行更新的，查阅相关API就可以知道：

通常，我们使用Hive来对一段时间内的数据进行分析和汇总，提交作业之后，如果是在Yarn上提交的，可以看到Yarn集群上在运行MapReduce作业

而Hbase则是我们为了交互而设计的，其响应速度比Hive快很多很多：

当然，提一句，现在的Hive可以依靠于Spark作为作业流引擎，速度要快一些：

75. 聊一聊MySQL和Hbase的区别

这个问题太泛泛，简单说几点：

MySQL是关系数据库，而HBase是非关系数据库
MySQL，就一张表来说，构建完毕之后，其基本结构不能再改变，其只能存储结构化的数据；而HBase的核心概念是列族，其可以管理很多很多的列，这一点比起MySQL要灵活的多
就容量来说，MySQL并不是天生支持分布式的，所以其存储空间容易受到机器限制，而HBase则不同，依靠于HDFS分布式文件系统，其存储容量可以达到无限大
就Null来说，Hbase是不会存储Null值的，而对于MySQL而言，必须要为空值分配存储空间
HBASE是Master/slave架构，启动之后我们可以看到，其存在HMaster和HRegionServer进程
Hbase必须依托于zookeeper来管理器元数据，而MySQL，当然不需要

76. 解释下 hbase 实时查询的原理

实时查询，可以认为是从内存中查询，一般响应时间在 1 秒内。HBase 的机制是数据先写入到内存中，当数据量达到一定的量（如 128M），再写入磁盘中， 在内存中，是不进行数据的更新或合并操作的，只增加数据，这使得用户的写操作只要进入内存中就可以立即返回，保证了 HBase I/O 的高性能。

77. 描述 Hbase 中 scan 和 get 的功能以及实现的异同

按指 定 RowKey 获 取 唯 一 一 条 记 录， get 方法 （ org.apache.hadoop.hbase.client.Get ） Get的方法处理分两种： 设置了ClosestRowBefore和没有设置的 rowlock 主要是用来保证行的事务性，即每个get 是以一个 row 来标记的.一个 row 中可以有很多 family 和 column。
按指定的条件获取一批记录，scan 方法(org.apache.Hadoop.hbase.client.Scan)实现条件查询功能使用的就是 scan 方式
scan 可以通过 setCaching 与 setBatch 方法提高速度(以空间换时间)；
scan 可以通过 setStartRow 与 setEndRow 来限定范围([start，end]start? 是闭区间，end 是开区间)。

范围越小，性能越高；
scan 可以通过 setFilter 方法添加过滤器，这也是分页、多条件查询的基础。 3.全表扫描，即直接扫描整张表中所有行记录。

78. Hbase 内部是什么机制?

在 HBase 中无论是增加新行还是修改已有的行，其内部流程都是相同的。HBase 接到命令后存下变化信息，或者写入失败抛出异常。默认情况下，执行写入时会写到两个地方：预写式日志（write-ahead log，也称 HLog）和 MemStore。HBase 的默认方式是把写入动作记录在这两个地方，以保证数据持久化。只有当这两个地方的变化信息都写入并确认后，才认为写动作完成。

MemStore 是内存里的写入缓冲区，HBase 中数据在永久写入硬盘之前在这里累积。当MemStore 填满后，其中的数据会刷写到硬盘，生成一个HFile。HFile 是HBase 使用的底层存储格式。HFile 对应于列族，一个列族可以有多个 HFile，但一个 HFile 不能存储多个列族的数据。在集群的每个节点上，每个列族有一个MemStore。大型分布式系统中硬件故障很常见，HBase 也不例外。

设想一下，如果MemStore 还没有刷写，服务器就崩溃了，内存中没有写入硬盘的数据就会丢失。HBase 的应对办法是在写动作完成之前先写入 WAL。HBase 集群中每台服务器维护一个 WAL 来记录发生的变化。WAL 是底层文件系统上的一个文件。直到WAL 新记录成功写入后，写动作才被认为成功完成。这可以保证 HBase 和支撑它的文件系统满足持久性。

大多数情况下，HBase 使用Hadoop分布式文件系统（HDFS）来作为底层文件系统。如果 HBase 服务器宕机，没有从 MemStore 里刷写到 HFile 的数据将可以通过回放 WAL 来恢复。你不需要手工执行。Hbase 的内部机制中有恢复流程部分来处理。每台 HBase 服务器有一个 WAL，这台服务器上的所有表（和它们的列族）共享这个 WAL。你可能想到，写入时跳过 WAL 应该会提升写性能。但我们不建议禁用 WAL， 除非你愿意在出问题时丢失数据。如果你想测试一下，如下代码可以禁用 WAL： 注意：不写入 WAL 会在 RegionServer 故障时增加丢失数据的风险。关闭 WAL， 出现故障时 HBase 可能无法恢复数据，没有刷写到硬盘的所有写入数据都会丢失。

79. HBase 宕机如何处理?

宕机分为 HMaster 宕机和 HRegisoner 宕机.

如果是 HRegisoner 宕机，HMaster 会将其所管理的 region 重新分布到其他活动的 RegionServer 上，由于数据和日志都持久在 HDFS 中，该操作不会导致数据丢失,所以数据的一致性和安全性是有保障的。

如果是 HMaster 宕机， HMaster 没有单点问题， HBase 中可以启动多个HMaster，通过 Zookeeper 的 Master Election 机制保证总有一个 Master 运行。即ZooKeeper 会保证总会有一个 HMaster 在对外提供服务。

80. HRegionServer宕机如何处理?

ZooKeeper 会监控 HRegionServer 的上下线情况，当 ZK 发现某个 HRegionServer 宕机之后会通知 HMaster 进行失效备援；
HRegionServer 会停止对外提供服务，就是它所负责的 region 暂时停止对外提供服务
HMaster 会将该 HRegionServer 所负责的 region 转移到其他 HRegionServer 上，并且会对

HRegionServer 上存在 memstore 中还未持久化到磁盘中的数据进行恢复;
这个恢复的工作是由 WAL重播 来完成，这个过程如下：
wal实际上就是一个文件，存在/hbase/WAL/对应RegionServer路径下
宕机发生时，读取该RegionServer所对应的路径下的wal文件，然后根据不同的region切分成不同的临时文件recover.edits
当region被分配到新的RegionServer中，RegionServer读取region时会进行是否存在recover.edits，如果有则进行恢复

81. hbase写数据 和 读数据过程

获取region存储位置信息
写数据和读数据一般都会获取hbase的region的位置信息。大概步骤为：

从zookeeper中获取.ROOT.表的位置信息，在zookeeper的存储位置为/hbase/root-region-server；
根据.ROOT.表中信息，获取.META.表的位置信息；
META.表中存储的数据为每一个region存储位置；
向hbase表中插入数据
hbase中缓存分为两层：Memstore 和 BlockCache

首先写入到 WAL文件 中，目的是为了数据不丢失；
再把数据插入到 Memstore缓存中，当 Memstore达到设置大小阈值时，会进行flush进程；
flush过程中，需要获取每一个region存储的位置。
从hbase中读取数据
BlockCache 主要提供给读使用。读请求先到 Memtore中查数据，查不到就到 BlockCache 中查，再查不到就会到磁盘上读，并把读的结果放入 BlockCache 。

BlockCache 采用的算法为 LRU（最近最少使用算法），因此当 BlockCache 达到上限后，会启动淘汰机制，淘汰掉最老的一批数据。

一个 RegionServer 上有一个 BlockCache 和N个 Memstore，它们的大小之和不能大于等于 heapsize * 0.8，否则 hbase 不能启动。默认 BlockCache 为 0.2，而 Memstore 为 0.4。对于注重读响应时间的系统，应该将 BlockCache 设大些，比如设置BlockCache =0.4，Memstore=0.39。这会加大缓存命中率。

82. HBase优化方法

优化手段主要有以下四个方面

减少调整
减少调整这个如何理解呢？HBase中有几个内容会动态调整，如region（分区）、HFile，所以通过一些方法来减少这些会带来I/O开销的调整

Region
如果没有预建分区的话，那么随着region中条数的增加，region会进行分裂，这将增加I/O开销，所以解决方法就是根据你的RowKey设计来进行预建分区，减少region的动态分裂。

HFile
HFile是数据底层存储文件，在每个memstore进行刷新时会生成一个HFile，当HFile增加到一定程度时，会将属于一个region的HFile进行合并，这个步骤会带来开销但不可避免，但是合并后HFile大小如果大于设定的值，那么HFile会重新分裂。为了减少这样的无谓的I/O开销，建议估计项目数据量大小，

给HFile设定一个合适的值

减少启停
数据库事务机制就是为了更好地实现批量写入，较少数据库的开启关闭带来的开销，那么HBase中也存在频繁开启关闭带来的问题。

关闭Compaction，在闲时进行手动Compaction

因为HBase中存在Minor Compaction和Major Compaction，也就是对HFile进行合并，所谓合并就是I/O读写，大量的HFile进行肯定会带来I/O开销，甚至是I/O风暴，所以为了避免这种不受控制的意外发生，建议关闭自动Compaction，在闲时进行compaction

批量数据写入时采用BulkLoad

如果通过HBase-Shell或者JavaAPI的put来实现大量数据的写入，那么性能差是肯定并且还可能带来一些意想不到的问题，所以当需要写入大量离线数据时建议使用BulkLoad

减少数据量
虽然我们是在进行大数据开发，但是如果可以通过某些方式在保证数据准确性同时减少数据量，何乐而不为呢?

开启过滤，提高查询速度
开启BloomFilter，BloomFilter是列族级别的过滤，在生成一个StoreFile同时会生成一个MetaBlock，用于查询时过滤数据

使用压缩：一般推荐使用Snappy和LZO压缩

合理设计
在一张HBase表格中RowKey和ColumnFamily的设计是非常重要，好的设计能够提高性能和保证数据的准确性

RowKey设计：应该具备以下几个属性
散列性：散列性能够保证相同相似的rowkey聚合，相异的rowkey分散，有利于查询
简短性：rowkey作为key的一部分存储在HFile中，如果为了可读性将rowKey设计得过长，那么将会增加存储压力
唯一性：rowKey必须具备明显的区别性
业务性：举些例子
假如我的查询条件比较多，而且不是针对列的条件，那么rowKey的设计就应该支持多条件查询
如果我的查询要求是最近插入的数据优先，那么rowKey则可以采用叫上Long.Max-时间戳的方式，这样rowKey就是递减排列
列族的设计

列族的设计需要看应用场景

多列族设计的优劣

优势：
HBase中数据时按列进行存储的，那么查询某一列族的某一列时就不需要全盘扫描，只需要扫描某一列族，减少了读I/O；
其实多列族设计对减少的作用不是很明显，适用于读多写少的场景。

劣势：
降低了写的I/O性能。原因如下：数据写到store以后是先缓存在memstore中，同一个region中存在多个列族则存在多个store，每个store都一个memstore，当其实memstore进行flush时，属于同一个region
的store中的memstore都会进行flush，增加I/O开销。

83. 为什么不建议在 HBase 中使用过多的列族

在 Hbase 的表中，每个列族对应 Region 中的一个Store，Region的大小达到阈值时会分裂，因此如果表中有多个列族，则可能出现以下现象：

一个Region中有多个Store，如果每个CF的数据量分布不均匀时，比如CF1为100万，CF2为1万，则Region分裂时导致CF2在每个Region中的数据量太少，查询CF2时会横跨多个Region导致效率降低。

如果每个CF的数据分布均匀，比如CF1有50万，CF2有50万，CF3有50万，则Region分裂时导致每个CF在Region的数据量偏少，查询某个CF时会导致横跨多个Region的概率增大。

多个CF代表有多个Store，也就是说有多个MemStore(2MB)，也就导致内存的消耗量增大，使用效率下降。

Region 中的 缓存刷新 和 压缩 是基本操作，即一个CF出现缓存刷新或压缩操作，其它CF也会同时做一样的操作，当列族太多时就会导致IO频繁的问题。

84. Hbase行键列族的概念，物理模型，表的设计原则?

行键：是hbase表自带的，每个行键对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分成多个region，每个region记录着行键的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

Rowkey的设计原则：各个列族数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少(按照列族进行存储，按照region进行读取，不必要的io操作)，经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

85. HBase的特点是什么?

(1)hbase是一个分布式的基于列式存储的数据库，基于hadoop的HDFS存储，zookeeper进行管理。

(2)hbase适合存储半结构化或非结构化数据，对于数据结构字段不够确定或者杂乱无章很难按一个概念去抽取的数据。

(3)hbase为null的记录不会被存储。

(4)基于的表包括rowkey，时间戳和列族。新写入数据时，时间戳更新，同时可以查询到以前的版本。

(5)hbase是主从结构。Hmaster作为主节点，hregionserver作为从节点。

86. 请描述如何解决Hbase中region太小和region太大带来的结果。

Region过大会发生多次compaction，将数据读一遍并写一遍到hdfs上，占用io，region过小会造成多次split，region会下线，影响访问服务，调整hbase.heregion.max.filesize为256m。

87. Hbase行健列族的概念，物理模型，表的设计原则?

行健：是hbase表自带的，每个行健对应一条数据。
列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区分。
物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。
rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。
列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。